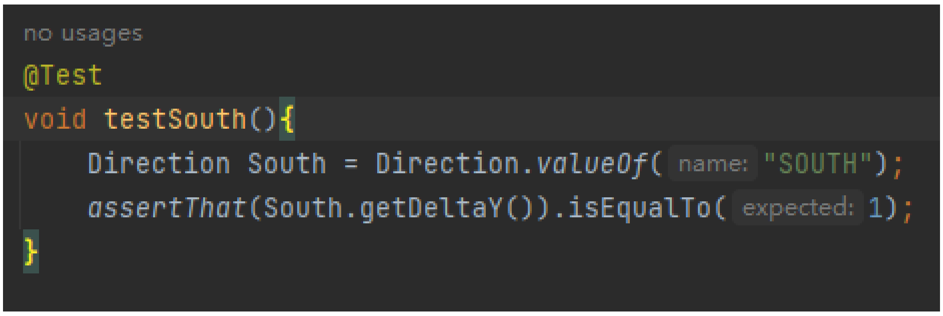
Junxian Liu
cs 472

UnitTesting Report

Task 2.1

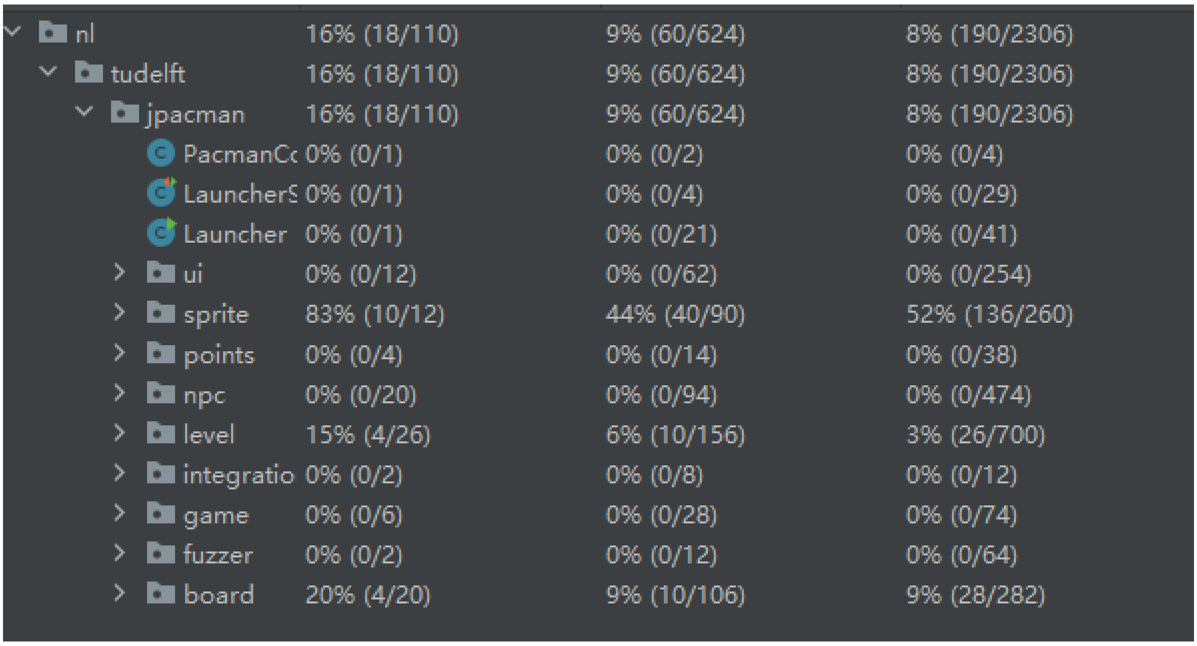
First:

SOUTH test code:



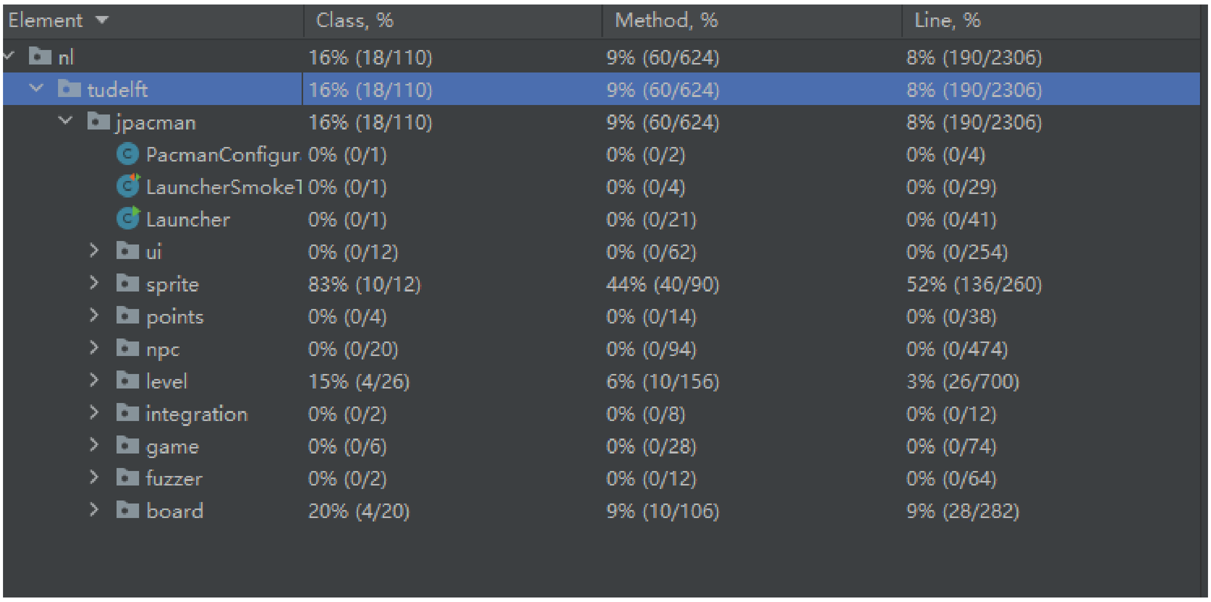
```
no usages
@Test
void testSouth(){
    Direction South = Direction.valueOf( name: "SOUTH");
    assertThat(South.getDeltaY()).isEqualTo( expected: 1);
}
```

Coverage before the test:



nl	16% (18/110)	9% (60/624)	8% (190/2306)
tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
PacmanCc	0% (0/1)	0% (0/2)	0% (0/4)
LauncherS	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
ui	0% (0/12)	0% (0/62)	0% (0/254)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
points	0% (0/4)	0% (0/14)	0% (0/38)
npc	0% (0/20)	0% (0/94)	0% (0/474)
level	15% (4/26)	6% (10/156)	3% (26/700)
integratio	0% (0/2)	0% (0/8)	0% (0/12)
game	0% (0/6)	0% (0/28)	0% (0/74)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
board	20% (4/20)	9% (10/106)	9% (28/282)

Coverage after the test:

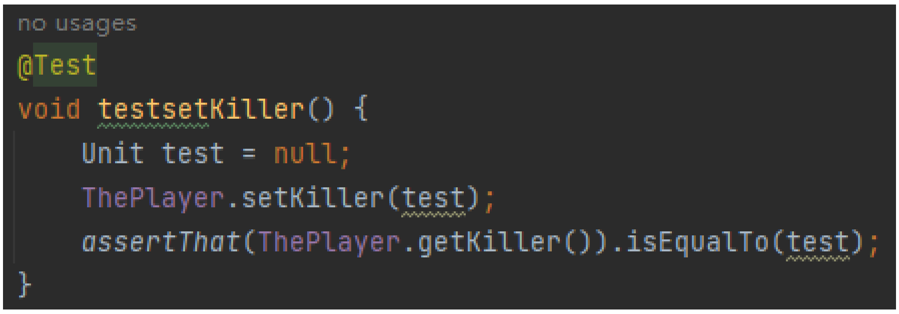


Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (60/624)	8% (190/2306)
tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
PacmanConfigur	0% (0/1)	0% (0/2)	0% (0/4)
LauncherSmoke1	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
ui	0% (0/12)	0% (0/62)	0% (0/254)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
points	0% (0/4)	0% (0/14)	0% (0/38)
npc	0% (0/20)	0% (0/94)	0% (0/474)
level	15% (4/26)	6% (10/156)	3% (26/700)
integration	0% (0/2)	0% (0/8)	0% (0/12)
game	0% (0/6)	0% (0/28)	0% (0/74)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
board	20% (4/20)	9% (10/106)	9% (28/282)

Since there was a north test before, the south test did not change the coverage anymore.

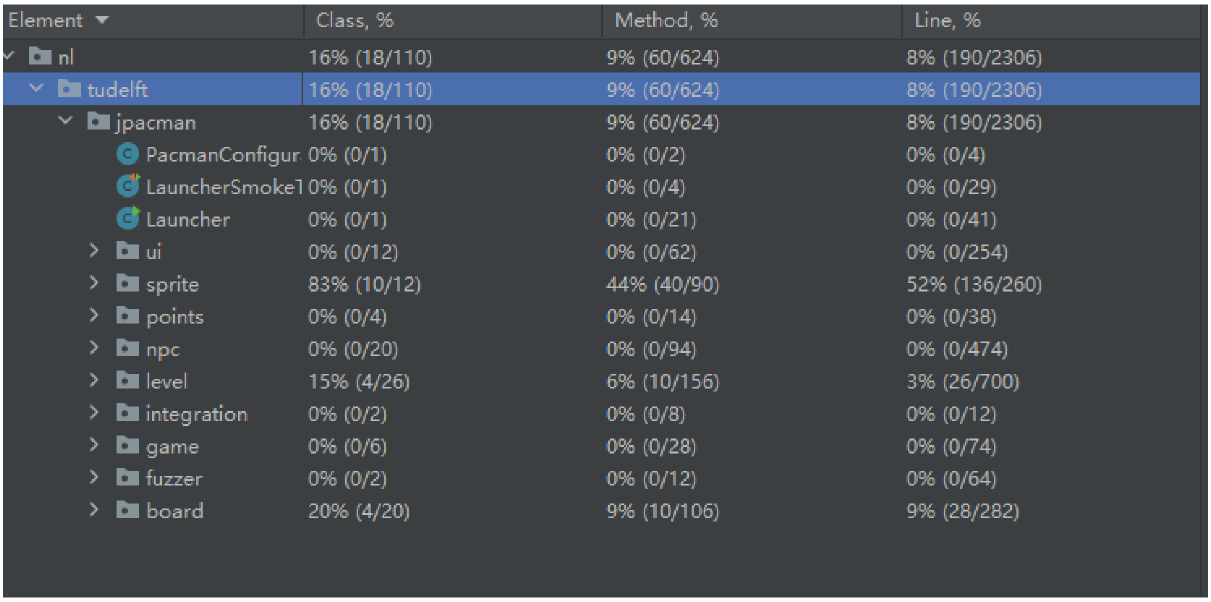
Second:

setkiller test code:



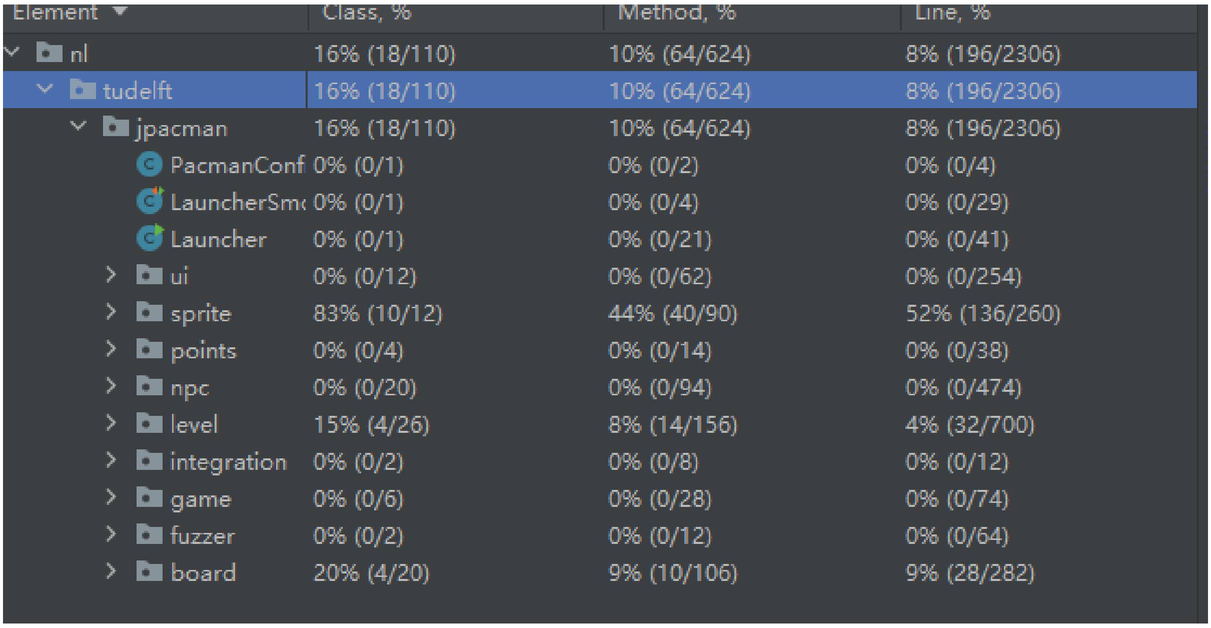
```
no usages
@Test
void testsetKiller() {
    Unit test = null;
    ThePlayer.setKiller(test);
    assertThat(ThePlayer.getKiller()).isEqualTo(test);
}
```

Before test:



Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (60/624)	8% (190/2306)
tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
PacmanConfigur	0% (0/1)	0% (0/2)	0% (0/4)
LauncherSmokeT	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
ui	0% (0/12)	0% (0/62)	0% (0/254)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
points	0% (0/4)	0% (0/14)	0% (0/38)
npc	0% (0/20)	0% (0/94)	0% (0/474)
level	15% (4/26)	6% (10/156)	3% (26/700)
integration	0% (0/2)	0% (0/8)	0% (0/12)
game	0% (0/6)	0% (0/28)	0% (0/74)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
board	20% (4/20)	9% (10/106)	9% (28/282)

After test:

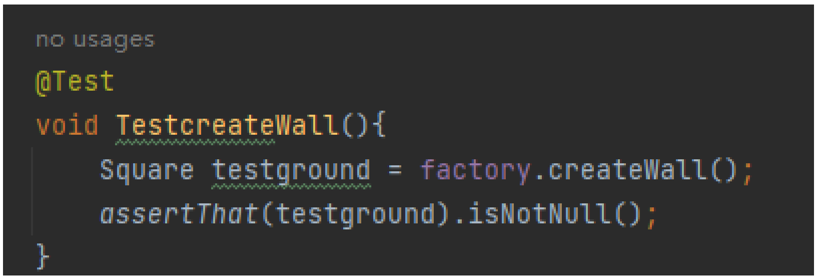


Element	Class, %	Method, %	Line, %
nl	16% (18/110)	10% (64/624)	8% (196/2306)
tudelft	16% (18/110)	10% (64/624)	8% (196/2306)
jpacman	16% (18/110)	10% (64/624)	8% (196/2306)
PacmanConf	0% (0/1)	0% (0/2)	0% (0/4)
LauncherSm	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
ui	0% (0/12)	0% (0/62)	0% (0/254)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
points	0% (0/4)	0% (0/14)	0% (0/38)
npc	0% (0/20)	0% (0/94)	0% (0/474)
level	15% (4/26)	8% (14/156)	4% (32/700)
integration	0% (0/2)	0% (0/8)	0% (0/12)
game	0% (0/6)	0% (0/28)	0% (0/74)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
board	20% (4/20)	9% (10/106)	9% (28/282)

After the setkiller test, the method percentage is increased to 10%.

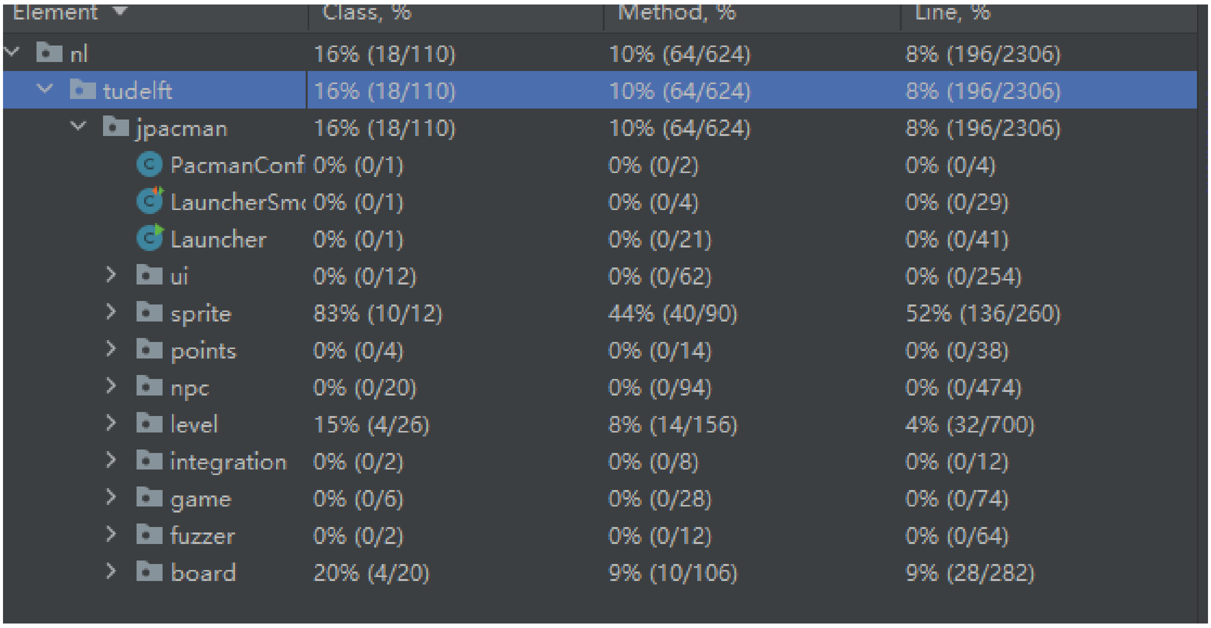
Third:

createWall test code:



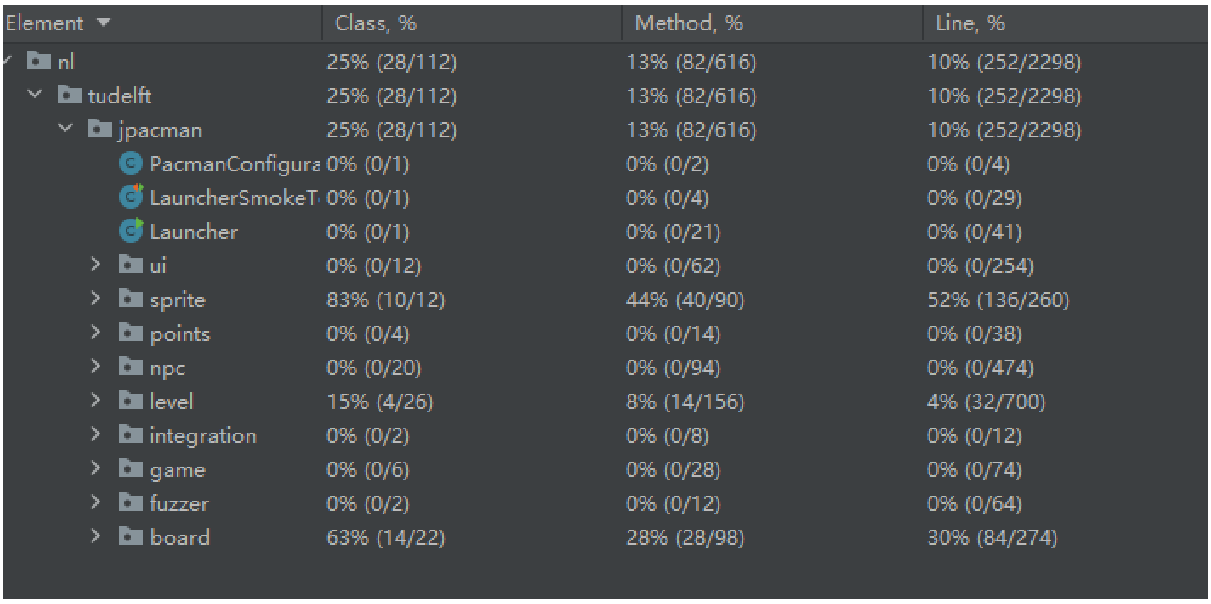
```
no usages
@Test
void TestcreateWall(){
    Square testground = factory.createWall();
    assertThat(testground).isNotNull();
}
```

Before test:



Element	Class, %	Method, %	Line, %
nl	16% (18/110)	10% (64/624)	8% (196/2306)
tudelft	16% (18/110)	10% (64/624)	8% (196/2306)
jpacman	16% (18/110)	10% (64/624)	8% (196/2306)
PacmanConf	0% (0/1)	0% (0/2)	0% (0/4)
LauncherSm	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
ui	0% (0/12)	0% (0/62)	0% (0/254)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
points	0% (0/4)	0% (0/14)	0% (0/38)
npc	0% (0/20)	0% (0/94)	0% (0/474)
level	15% (4/26)	8% (14/156)	4% (32/700)
integration	0% (0/2)	0% (0/8)	0% (0/12)
game	0% (0/6)	0% (0/28)	0% (0/74)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
board	20% (4/20)	9% (10/106)	9% (28/282)

After test:



Element	Class, %	Method, %	Line, %
nl	25% (28/112)	13% (82/616)	10% (252/2298)
tudelft	25% (28/112)	13% (82/616)	10% (252/2298)
jpacman	25% (28/112)	13% (82/616)	10% (252/2298)
PacmanConfigura	0% (0/1)	0% (0/2)	0% (0/4)
LauncherSmokeT	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
ui	0% (0/12)	0% (0/62)	0% (0/254)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
points	0% (0/4)	0% (0/14)	0% (0/38)
npc	0% (0/20)	0% (0/94)	0% (0/474)
level	15% (4/26)	8% (14/156)	4% (32/700)
integration	0% (0/2)	0% (0/8)	0% (0/12)
game	0% (0/6)	0% (0/28)	0% (0/74)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
board	63% (14/22)	28% (28/98)	30% (84/274)

After the createWall test, the Class is increased to 25%, the Method is increased to 13%, and Line is increased to 10%.

TASK 3

Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?

They are not similar, I think it is because their counting way is different, the total Method in IntelliJ for level folder is 156, and JaCoCo is 69.

Did you find helpful the source code visualization from JaCoCo on uncovered branches?

Yes, it shows which branches have the most coverage rate.

Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?

I would prefer IntelliJ's coverage window, because I can see which file belongs to which folder, and IntelliJ is more concise, and JaCoCo gives me a lot of information at the first time.

My Github Link: https://github.com/WuliuQ/cs472-team6